Creative Technologies in the Classroom: Programación dinámica, variables de tiempo, uso de imágenes, bucle `for` y condicional `if`

Colegio Seminario Diocesano de Duitama - M&T

1. Movimiento

La programación dinámica en `processing` no genera movimiento real de los elementos dibujados, el bucle `draw` dibuja n veces (definidas en 60 veces por segundo por defecto) los objetos, esto indica que el movimiento de los objetos en `processing` no es más que una simulación de movimiento lograda con un simple efecto conseguido con la combinación de instrucciones de dibujo y definición del `background` del canvas; en este ejemplo se desarrollará un reloj análogo en que se simulará el movimiento de las manesillas del reloj.

Lo primero es entender cómo determinar las coordenadas polares de un punto que se mueve sobre una circunferencia como en la Figura 1; debido a que es más fácil considerar la magnitud del ángulo central α, las coordenadas del punto P se toman en relación a éste ángulo, son las llamadas *coordenadas polares de* P.

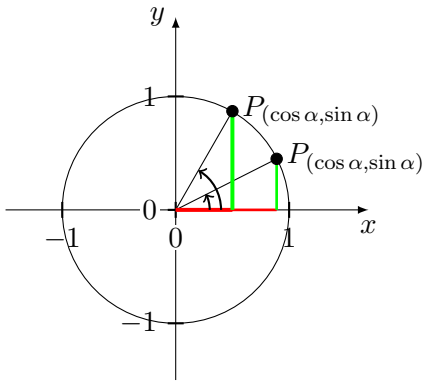

Figura 1: Movimiento del punto P sobre la circunferencia unitaria

Puede calcularse entonces la magnitud del ángulo que recorre el extremo de cualquier manecilla del reloj, y debido a que uno de los extremos siempre está fijo al centro del tablero del reloj, las coordenadas del otro extremo se puden establecer usando las coordenadas polares de tal punto, así dado que hay 60 segundos en un minuto (una revolución), el ángulo que recorre el extremo de la manecilla del segundero cada segundo es

$$\alpha = \frac{2\pi}{60} = \frac{\pi}{30}$$

luego para simular el movimiento del segundero, se dibujará una línea (segmento) cada segundo con un de sus extremos el centro del tablero y el otro extremo en el punto P desplazado $s\alpha$ veces, donde s es el número actual de segundos; es importante notar que cuando $s = 0$,

$\cos(s\alpha) = 1$ y $\sin(s\alpha) = 0$ lo que indica que la manecilla estaría sobre el semieje positivo de abscisas (ver Figura 1) pero en un reloj cuando $s = 0$ la manecilla debe estar sobre el semieje positivo de ordenadas, por tanto al ángulo $s\alpha$ hay que restarle $\frac{\pi}{2}$ para conseguir la posición correcta de la manecilla.

```
/*
Variables enteras que definiran la longitud de la manecilla
segundero, el centro del tablero y el segundo actual en el reloj
interno del sistema
*/
int radioS, h, k, s;

/*
Variable decimal correspondiente al angulo que recorre el segundero
en s segundos
*/
float angleS = 0;

void setup(){
  size(300, 300);
  radioS = 100; // Longitud del segundero
  h = width / 2; // width determina el ancho del canvas
  k = height / 2; // height determina la altura del canvas
}

void draw(){
  s = second(); // second() determina el segundo actual
  angleS = s * PI / 30 - PI / 2;
  line(h, k, h + radioS * cos(angleS), k + radioS * sin(angleS));
}
```

La ejecución del anterior código resulta en 60 líneas como se muestra en la Figura 2

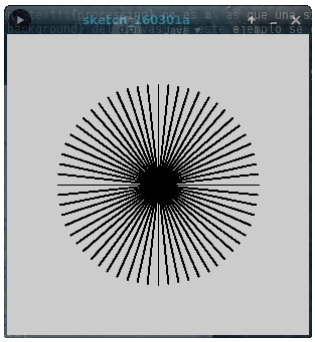

Figura 2: Sesenta líneas de longitud 100 dibujadas una cada segundo

para conseguir el efecto de movimiento simplemente se dibuja el `background` antes de dibujar cada nueva línea, esto puede hacerse incluyento la línea

```
background(120);
```

entre las líneas 23 y 24, con lo cual se obtiene un fondo de color gris dibujado justo antes de dibujar la siguiente línea o segmento, esto oculta todas las líneas anteriores dejando ver únicamente la correspondiente línea del segundo actual.

1.1. Hazlo tu

Completa las manecillas del reloj, necesitaras agregar dos variables por cada manecilla (minutero y horario), una variable de tipo decimal (`float`) para el ángulo que recorre la manecilla en cada movimiento (minuto u hora) y una variable de tipo entero (`int`) para determinar la longitud de la manecilla; ten presente que la manecilla horaria recorre un ángulo diferente a las otras dos manecillas, tendrás que calcularlo. Para determinar el minuto actual en el reloj del sistema utilizas la variable `minute()`, y para la hora actual `hour()`. Utiliza lo que ya sabes de color para que cada manecilla sea de un color diferente y grozor de línea también diferente, que cada manecilla se identifíque claramente, recuerda que la más larga de las tres es el segundero y la más corta la manecilla horaria.

2. El bucle `for`

```
for (tipo indice = inicio; final; incremento){
  Instrucciones a ejecutar indice veces;
}
```

El bucle `for` ejecuta un número (índice) determinado de veces una serie de instrucciones, su sintaxis incluye una variable que puede ser de tipo entero (`int`) o decimal `float` que hace las véces de índice o contador de repeticiones, esta variable tiene un valor inicial, un valor final y un valor de incremento dependiendo de si la cuenta es de menor a mayor o viceversa. e.g.

```
for (int i = 1; i < 11; i++){
  println(i);
}
```

imprime en la consola los números enteros de 1 a 10.

2.1. Las divisiones del reloj

Utilizando un blucle for, se van a dibujar las divisiones horarias en el tablero del reloj, cada división será una línea (segmento) de longitud 30 pixels y dado que cada división corresponde a una hora, éstas están separadas $\frac{\pi}{6}$ una de otra.

Incluyendo el siguiente bucle `for` dentro del `draw`, al final del código que define las manecillas del reloj se consigue un resultado como el observado en la figura 3.

```
  for (float i = 0; i < 2 * PI; i += PI / 6){
    line(h + 100 * cos(i), k + 100 * sin(i), //
    h + 130 * cos(i), k + 130 * sin(i));
  }
```

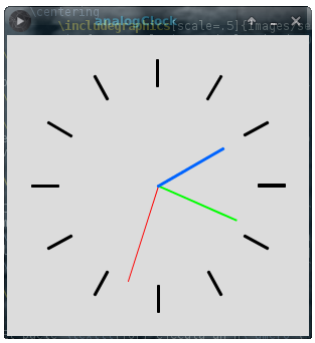

Figura 3: Reloj análogo con divisiones

Sugerencia: Ahora puedes intentar crear tus propias divisiones para el reloj, por ejemplo agregar las divisiones de cada minuto, deberías hacerlo con una longitud más corta y utilizando el mismo ángulo utilizado en la manecilla del minutero.

3. Variación de color

Como paso final para mejorar la apariencia del reloj se construirá un efecto de variación de color que hará que el color del tablero y las divisiones varíe en función de la hora del día, en la noche el tablero será oscuro y las divisiones claras y en el día el tablero se aclarará a medida que amanece y se oscurecerá progresivamente después del medio día, el color de las divisiones permanecerá siempre en contraste con el color del tablero.

Lo primero es entender el modelo matemático de variación de la luz a través del día, el cual puede ser aproximado por la función sinusoidal

$$f(t) = -255\cos\left(\frac{\pi}{12}t\right) \tag{1}$$

donde t esta dado en horas. Esta ecuación describe la curva

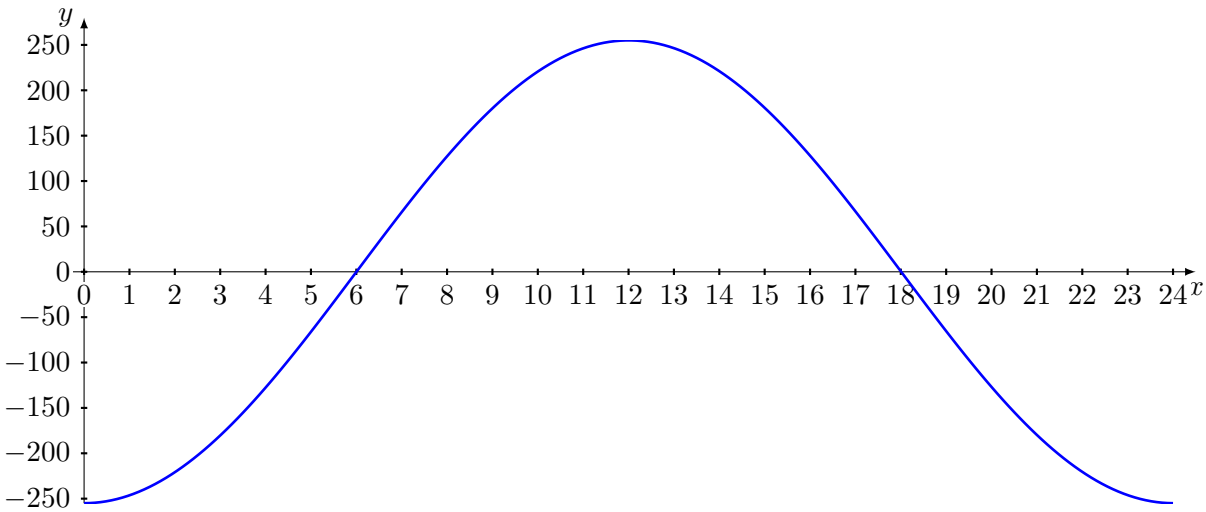

Figura 4: Descrición gráfica del cambio de iluminación en 24 horas

En la Figura 4 se representa un periodo completo de la curva sinusoidal con el eje x en el intervalo $[0, 24]$ horas y el eje y en el rango $[-255, 255]$ ya que la asignación de color RGB esta

dada en un rango de $[0, 255]$ (los valores negativos se tomarán como cero), la parte interesante de la curva, que refleja el cambio en la intensidad del color (luz) a medida que transcurre el día se encuentra entre 6 y 18 (en el eje horizontal), en este intervalo se ve que de 6 (am) a 12 (m) la intensidad del color (luz) va en aumento y de 12 (m) a 18 esta disminuyendo, en los intervalos de 0 a 6 y de 18 a 24 hay oscuridad luego el color debe ser negro.

3.1. Condicional `if`

```
if (condicion){
  instrucciones si condicion es verdadera;
}
else{
  instrucciones si condicion es falsa;
}
```

Una función `if`, *condiciona* la ejecución de una serie de instrucciones al cumplimiento de una o varias condiones que se determinan en el parámetro de entrada `condicion` a través de los operadores condicionales o booleanos e.g.

```
if (x == 1 && y == 0){
  println("x es uno e y es cero");
}
```

en este ejemplo se comparan las variables x e y con los número 1 y 0 respectivamente y si a la vez $x = 1$ y $y = 0$ se imprime un mensaje en la consola.

La función condicional `if` se usará para hacer que el `background` cambie progresivamente de negro a blanco y viceversa a medida que transcurre el día, para esto es suficiente reemplazar la línea que define el `background` a gris por el código

```
if (hour() >= 6 && hour() <= 18){
  background(-255 * cos(PI / 12 * hour()));
}
else{
  background(0);
}
```

en el cual se utiliza la ecuación (1) para que si la hora (hour()) esta en el intervalo $[6, 18]$ cámbie (cada hora) de gris a blanco (hasta las 12 m) y de blanco a gris (hasta las 18), si la hora es menor a 6 o mayor a 18 el `background` se configura en color negro.

3.2. Hazlo tu

Utilizando la ecuación (1) (modificada) y la función condicional `if` genera un cambio progresivo en el color de las divisiones del reloj para que permanezcan en contraste con el `background` (tablero oscuro divisiones blancas y viceversa) a medida que transcurre el día.

4. Incluir imágenes

Las imágenes en `processing` son una herramienta que permite mejorar la apariencia de los programas además de las funciones propias para el tratamiento de imágenes que se estudianrán más adelante en este mismo curso.

El tipo de dato de `processing`, `PImage` permite cargar archivos de imágen en una variable, ver `PImage en reference/processing`,

Para poder utilizar imágenes en un sketch de processing primero deben cargarse dentro del directorio del sketch después de guardarlo, hacer esto es tan fácil como llevar los archivos de imagen desde el directorio contenedor a la ventana de processing, como se indica en la Figura 5

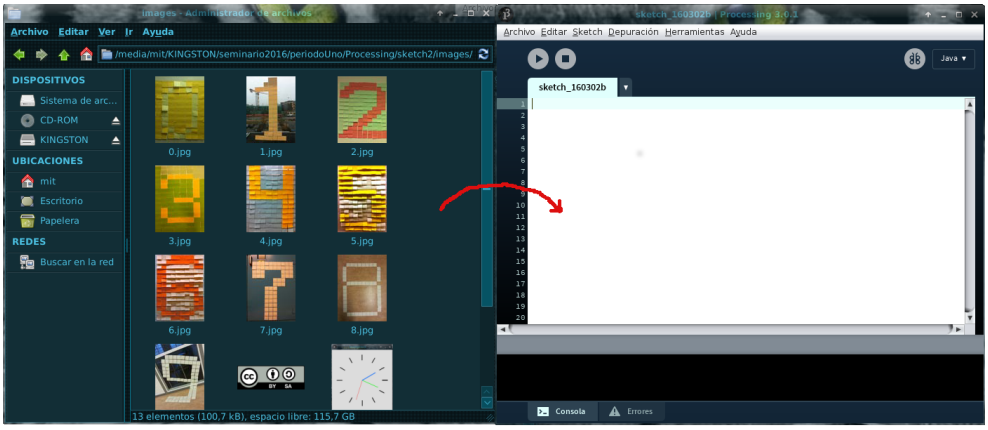

Figura 5: Cargar imágenes en un sketch de processing

al llevar los archivos de imagen sobre la ventana de processing se creará un subdirectorio `data` en el directorio del sketch que contiene las imágenes cargadas tal como se muestra en la Figura 6

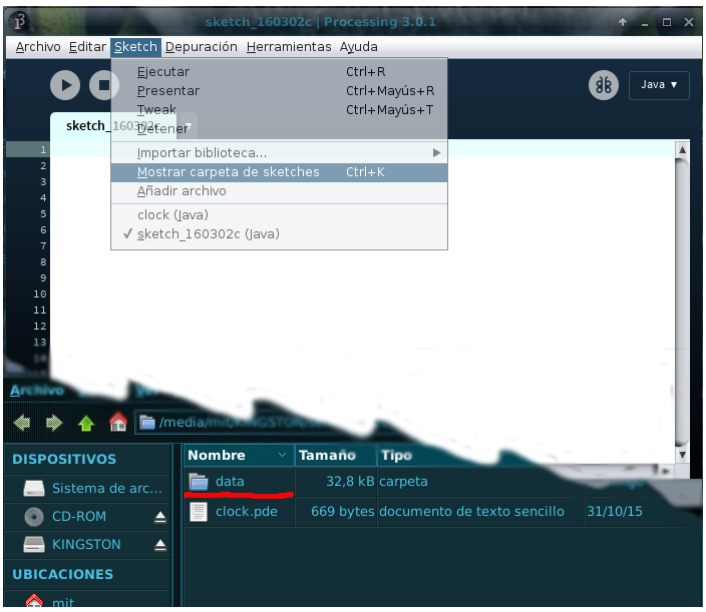

Figura 6: Carpeta dada con las imánes incluidas en el sketch

4.1. Reloj digital

Para construir un reloj digital se utilizarán 10 imágenes, cada una con uno de los números dígitos, éstas serán “almacenadas” en un vector de tipo `PImage` tal como se muestra en el siguiente código

```
// Genera un vector tipo PImage que almacenara las imagenes
PImage im[] = new PImage[10];

// Genera un vector de tipo cadena (texto) que contiene los nombres de los
    archivos de imagen
String imFile[] = {"0.jpg", "1.jpg", "2.jpg", "3.jpg", "4.jpg", "5.jpg", //
"6.jpg", "7.jpg", "8.jpg", "9.jpg"};

void setup(){
  size(140, 285);

  // carga en el vector im (de tipo PImage) cada una de las imagenes
  for (int i = 0; i < 10; i++){
    im[i] = loadImage(imFile[i]);
  }
}

void draw(){
  int s = second();
  int s_dec = int(s / 10); // determina el digito de las decenas
  int s_uni = s - s_dec * 10; // determina el digito de las unidades

  image(im[s_dec], 0, 190); // carga en el canvas el digito de las decenas
  image(im[s_uni], 70, 190); // carga en el canvas el digito de las unidades
}
```

4.1.1. Hazlo tu

El reloj tiene hasta ahora únicamente los dígitos que muestran los segundos, contruye las variables necesarias y carga las imágenes correspondientes a los dígitos de las horas y los minutos.

El resultado que debes conseguir es tal como el mostrado en la figura

Figura 7: Reloj Digital

5. Programa interactivos

El desarrollo de sketches interactivos en processing es una parte de este curso que se retomará a profundidad más adelante, por ahora se desarrollará una breve introducción, utilizando imágenes para simular botones interactivos, en este ejercicio se utilizará un botón circular, para ello lo siguiente es entender cómo se detectará el clic del mouse sobre el botón.

La función `mousePressed()` detecta que uno de los botones del mouse ha sido preciona-do.

Las variables de sistema `mouseX` y `mouseY` leen las coordenadas del puntero del mouse dentro del canvas.

Para determinar que se ha activado el botón, en este caso, un botón circular, es necesario asegurarse que el puntero del mouse se encuentre ubicado sobre el botón y no fuera de su perímetro, para eso, en el caso de un botón circular, es necesario tener presente la ecuación que describe una circunferencia

$$(x-h)^2 + (y-k)^2 = r^2 \tag{2}$$

en (2), h y k representan las coordenadas x e y del centro de la circunferencia y r el radio de la misma (mayores detalles sobre la ecuación de una circunferencia se estudian en un curso básico de geometría analítica).

Además se utiliza una variable de tipo `boolean` que adopta un valor lódigo `true` o `false` para determinar el estado del botón, activado = true, desactivado = false.

El siguiente código muestra un ejemplo de un botón que cambia el background del canvas.

```
int h, k, diameter;
boolean buttonState = false; // variable logica que determina el estado del
    boton
PImage button;

void setup(){
  size(300, 300);
  button  = loadImage("button.png");
  h = width / 2;
  k = height / 2;
  diameter = 100;
}

void draw(){
  if (buttonState == true){
    background(255);
  }
  else {
    background(0);
  }

  image(button, 100, 100, diameter, diameter);
}
```

```
// Acciones cuando se da click sobre la imagen del boton
void mousePressed(){
  if (sqrt(sq(mouseX - h) + sq(mouseY - k)) < diameter / 2 && //
  buttonState == false){
    buttonState = true;
  }
  else if (sqrt(sq(mouseX - h) + sq(mouseY - k)) < diameter / 2 && //
  buttonState == true){
    buttonState = false;
  }
}
```

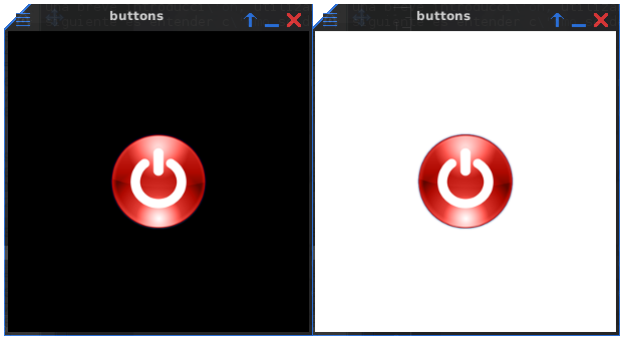

Figura 8: Ejemplo de botón

5.1. Hazto tu

Ahora que ya has visto como utilizar el mouse para desarrollar programas interactivos puedes intentar desarrollar algún programa que integre todo lo que has aprendido hasta ahora, por ejemplo intenta hacer un reloj que cambie entre anólogo y digital utilizando un botón, o que cámbie los colores de las manecillas del reloj análogo, o que cambie las imágenes que representan los números en el reloj digital, etc. son muchas las ideas que puedes desarrollar para aprovechar lo que ya sabes de programación en processing.

Lic. Fausto Mauricio Lagos Suárez
Mg. Ingeniería computacional y matemática